A STATE-SPACE FAULT MONITOR ARCHITECTURE AND ITS APPLICATION TO THE CASSINI SPACECRAFT

Glenn A. Macala
Jet Propulsion Laboratory
California Institute of Technology
M.S. 198-326, 4800 Oak Grove Drive
Pasadena, California 91109-8099
(818) 354-6398

Abstract

The Cassini spacecraft fault protection design includes fault monitors that use a state-space architecture to monitor the performance of spacecraft functions. This paper reports on the design of these monitors.

1 Introduction

The Cassini spacecraft fault protection design includes fault monitors that are designed to act as "idiot lights". They watch certain variables in the flight software and provide information as to the quality of operation of a function. The quality is described as a color: Black, Green, Yellow, or Red.

A monitor output of Black indicates that no information is available from a monitor: either the hardware being monitored is powered off or the function being monitored is not active at the time. A monitor output of Green indicates that the monitored variable is behaving in the expected manner and no action from fault protection is necessary. A monitor output of Yellow indicates that the monitored variable is not quite behaving in its expected manner, but it also is not behaving in a way that would require immediate action from fault protection. Lastly, a monitor output of Red indicates that the monitored variable is behaving in a manner that indicates a definite problem and that immediate action is required from fault protection.

2 State-Space Opinion Generation

The state-space monitor performs filtering on a variable that renders information on the performance of the function being monitored. The filtering creates both a test variable and its rate of change: $(\varepsilon,\dot{\varepsilon})$.

The following equations illustrate the formation of the filtered test variable and its rate of change.

$$\Delta\varepsilon = (\varepsilon_{raw} - \varepsilon)*(1\text{-gain}) \tag{2.1}$$

$$\dot{\varepsilon} = \Delta\varepsilon/\Delta T \tag{2.2}$$

$$\varepsilon = \varepsilon + \Delta\varepsilon \tag{2.3}$$

The raw test variable is called ε_{raw} and would typically be a quantity such as spacecraft attitude error as determined by the attitude controller. The filtered test variable, ε, is simply the result of passing the raw variable through a first-order digital filter whose update rate is ΔT. This filter incorporates raw information into the filtered value using a factor of (1-gain). The value of gain ranges from 0 to 1 depending upon the degree of filtering desired.

A test is then performed on this information resulting in an opinion of the performance of the monitored function. The opinion can be one of the following: *expected*, performance is nominal; *tolerable*, performance is off nominal, but is still within acceptable bounds; *unacceptable*, performance is degraded enough to require a fault protection response if it persists.

Figure 2.1 graphically illustrates the method used to determine monitor opinions.

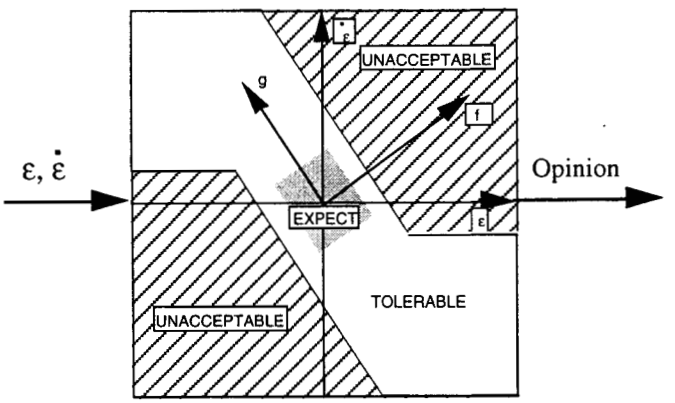

Figure 2.1 Phase Plane Regions For Opinion Generation

If, for example, attitude error was the monitored variable and $(\varepsilon,\dot{\varepsilon})$ was contained within the region marked *expected*, we note that attitude error is small and has a small rate of change. The monitor would thus issue an opinion of "expected". If $(\varepsilon,\dot{\varepsilon})$ was found to be in the region marked *unacceptable*, we note that attitude error is large and its rate of change is either making it grow larger or at least not reducing it fast enough. The monitor would issue an opinion of "unacceptable". Lastly if $(\varepsilon,\dot{\varepsilon})$ was contained in the region marked *tolerable*, we note that attitude error is larger than we would expect for nominal operation, but it is

reducing at an acceptable rate. The monitor would therefore issue an opinion of "tolerable".

The filtered test variable and its rate of change are transformed into a rotated coordinate frame (f,g axes in Figure 2.1) in order to more easily determine the region that $(\varepsilon,\dot{\varepsilon})$ currently lies in. The transformation is defined by the following equations.

$$f = k_1 * \varepsilon + k_2 * \dot{\varepsilon} \tag{2.4}$$

$$g = -k_2 * \varepsilon + k_1 * \dot{\varepsilon} \tag{2.5}$$

The tests to determine monitor opinion now become:

```
opinion = "tolerable"

if (|f| < f_small) & (|g| < g_small)
    opinion = "expected"
elseif |f| > f_large
    if (sign(f) > 0) & (ε̇ > -ε̇_small)
        opinion = "unacceptable"
    elseif (sign(f) < 0) & (ε̇ < ε̇_small)
        opinion = "unacceptable"
    end if
end if
```

3 Color Generation

A monitor's color is generated by using the sequence of opinions from a monitor to generate persistence counts. Two type of counters are used: a red counter and a green counter.

The red counter counts up every time the monitor outputs an opinion of "unacceptable". It counts down otherwise. Similarly, the green counter counts up every time the monitor outputs an opinion of "expected". It counts down otherwise. The red and green counters are limited to maximum values of redLimit and greenLimit, respectively. In addition, the counters cannot decrement below zero.

The red counter is implemented using the following logic:

```
if opinion = "unacceptable"
    redCount = redCount + redInc
else
    redCount = redCount - redDec
end if

if redCount > redLimit
    redCount = redLimit
elseif redCount < 0
    redCount = 0
endif
```

The green counter is implemented using the same type of logic:

```
if opinion = "expected"
    greenCount = greenCount + greenInc
else
    greenCount = greenCount - greenDec
end if

if greenCount > greenLimit
    greenCount = greenLimit
elseif greenCount < 0
    greenCount = 0
end if
```

The color output of the monitor is now determined by Table 3.1 *unless* the monitor's current opinion is "No Opinion". For that case, the monitor's color is always Black and all counters are reset to zero.

Table 3.1 Output Color Logic

redCount	greenCount	Output Color
0	greenLimit	Green
0	< greenLimit	Yellow
> 0, < redLimit	greenLimit	Yellow
> 0, < redLimit	< greenLimit	Yellow
redLimit	"any value"	Red

4 Monitor Parameters

The length of this paper precludes an in-depth discussion of the selection of a state-space monitor's parameters. The following is a brief summary of the parameters that must be selected and their primary effect on monitor behavior.

Equation (2.1) uses a low pass filter gain. The primary use of this gain is to limit noise that may be in the test signal and also to limit the noise present in the derived rate of change of the test signal, $\dot{\varepsilon}$.

Equations (2.4) & (2.5) use parameters k_1 and k_2. The primary effect of these gains is to adjust the rotation of the (f,g) coordinate system with respect to the $(\varepsilon,\dot{\varepsilon})$ system of Figure 2.1. These gains can be used to adjust such things as the maximum "expected" amount of error when error rate is zero and the maximum "expected" amount of error rate when error is zero.

The opinion logic of Section 2 uses parameters f_small and g_small. The primary effect of these parameters is to define the *expected* region of the monitor. Also used are parameters f_large and $\dot{\varepsilon}$_small. The primary effect of these parameters is to define the *unacceptable* regions of the monitor.

The color logic of Section 3 uses parameters redLimit, redInc, and redDec. The primary effect of these parameters is on the persistance counter used to trigger a fault response. When the redCount reaches redLimit, the monitor color turns Red and fault protection takes action. The separate increment and decrement parameters allow the designer to

adjust the rate at which a fault is declared independently of the allowable "recovery" time for "unacceptable" behavior.

The color logic of Section 3 also uses greenLimit, greenInc, and greenDec. The primary effect of these parameters is on the persistence counter used to signal nominal performance. When the greenCount reaches greenLimit, the monitor color will turn Green. This says that "expected" performance has been occuring quite regularly. Again, separate increment/decrement values allow the designer to adjust the amount of time "expected" behavior should persist independently of the penalty for time when simply "tolerable" or "unacceptable" behavior persists.

In general, each monitor's parameters must be set on a case by case basis through analysis of the characteristics and behavior of the function being monitored.

5 Monitor Telemetry

Telemetry relevant to the state-space fault monitor consists of "high water" marks. The "high water" marks tell flight operations just how bad monitored variables have ever gotten and also how long they stayed at non-nominal values.

Two "high water" marks are used on the state-space monitor's transformed variables: f and g. These "test variable" high water marks show just how large a monitored variable ever got. The following logic illustrates the formation of this telemetry:

```
if |f| > |f_highWater|,
    f_highWater = f
end if

if |g| > |g_highWater|
    g_highWater = g
end if
```

Two other "high water" marks are formed using the monitor's red and green counters. The logic for the formation of this telemetry follows:

```
if redCount > red_highWater,
    red_highWater = redCount
end if

if notGreen_count > notGreen_highWater
    notGreen_highWater = notGreen_count
end if
if Color ≠ Green
    notGreen_count = notGreen_count + 1
else
    notGreen_count = 0
end if
```

The red_highWater "persistence" high water mark shows just how close the monitor ever came to tripping a fault response (outputting a color of Red). The notGreen_highWater "persistence" high water mark shows the longest time period that a non-nominal condition existed in the monitored variable.

This telemetry is used by flight operations to analyze any faults that may occur. Another important use of this telemetry is to allow flight operations to anticipate, through trend analysis, possible future problems.

6 Application to Cassini

The state-space fault monitor is used on the Cassini spacecraft to monitor a number of controller error signals.

At certain times the attitude control of Cassini is performed using a Reaction Control Subsystem (RCS) consisting of hydrazine thrusters. At other times, attitude control is performed using a Reaction Wheel Assembly (RWA). During main engine burns, attitude control is performed using a gimbaled main engine and the RCS. All of these control systems are monitored by a state-space fault monitor that uses attitude error as its test variable: attitude error should be kept small.

Additionally, the RWA can run in a speed control mode while the RCS controls spacecraft attitude. A control loop drives the wheels to their commanded speeds using the RWA motor torque. A state-space monitor uses the speed error of this controller as its test variable: wheel speed error should be kept small.

Another Cassini application of the state-space monitor is the RCS rate controller. At certain times the RCS is used to control spacecraft angular rate. A state-space monitor is uses the controller's rate error as its test variable: spacecraft rate error should be kept small.

For a complete description of the design of Cassini fault monitors see [1].

References

[1] *Cassini AACS Fault Protection Design Document*, Internal Document, Jet Propulsion Laboratory , 1997.

Acknowledgments

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration. The author wishes to acknowledge that the work summarized in this paper was the result of a collaboration of a team of individuals that included D. Bernard, M. Brown, A. Lee, G. Macala, and R. Rasmussen.